UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/870,087	05/30/2001	Navin Kumar Sinha	JP920010012US1	6782

39903 7590 11/01/2005

ANTHONY ENGLAND
PO Box 5307
AUSTIN, TX 78763-5307

EXAMINER	
YIGDALL, MICHAEL J	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 11/01/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/870,087	SINHA, NAVIN KUMAR	
	Examiner	Art Unit	
	Michael J. Yigdall	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *19 August 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on August 19, 2005 has been entered. Claims 1-20 are pending.

Response to Arguments

2. Applicant's arguments have been considered but are moot in view of the new ground(s) of rejection.

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1-20 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 5,797,013 to Mahadevan et al. (art of record, "Mahadevan") in view of U.S. Patent No. 5,862,384 to Hirai (now made of record, "Hirai") in view of U.S. Patent No. 6,748,589 to Johnson et al. (now made of record, "Johnson").

With respect to claim 1 (currently amended), Mahadevan discloses a method of optimizing compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs (see, for example, the abstract), the method comprising the steps:

(1) providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* (see, for example, column 6, lines 22-27, which shows providing loop code for a loop written in a high-level language, and column 7, lines 30-35, which shows that the loop is iteratively executed some number of times, i.e. *n* times).

Mahadevan does not expressly disclose the limitation wherein the non-optimized loop code segment includes a call to a procedure.

However, Hirai discloses an analogous method of optimizing a loop code segment (see, for example, the abstract), and discloses a non-optimized loop code segment that includes a call to a procedure (see, for example, loop block 1105 in FIG. 11, which includes calls 1108 and 1109 to procedures).

Mahadevan also discloses the steps:

(2) providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for causing execution of the loop construct the loop repetition number of times *n* (see, for example, column 7, lines 35-39, which shows providing a variable that has values satisfying initial and terminating conditions for executing the loop *n* times).

Although Hirai discloses that the procedure includes ones of the program variables (see, for example, procedure 1118 in FIG. 11, which shows that the procedure includes ones of the program variables 1102 used in the loop block 1105), Mahadevan and Hirai do not expressly disclose the limitation wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables.

However, Johnson discloses an analogous method of optimizing a loop code segment (see, for example, column 7, lines 20-27), and suggests that a call to a procedure, such as in Hirai, could invoke the procedure with arguments that include ones of the program variables used in the loop code segment (see, for example, column 7, lines 28-40, which shows an argument x to some function f, where x is a variable that has values satisfying initial and terminating conditions for the loop).

Mahadevan also discloses the steps:

(3) optimizing the non-optimized loop code segment for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times n (see, for example, column 10, lines 53-61, which shows optimizing the loop to provide code corresponding to the trip count, i.e. the number of repetitions n).

Although Mahadevan discloses that branches are omitted from the consolidated loop code segment to save CPU cycles and reduce branch mispredictions (see, for example, column 6, lines 28-32), Mahadevan does not expressly disclose the limitation wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and wherein the call is omitted from the consolidated

loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables.

However, Hirai further discloses that the consolidated loop code segment includes certain code of the non-optimized loop code segment and omits other invariant code (see, for example, FIG. 7 and column 23, lines 61-67), which is omitted if the call invoking the procedure results in no change in the program variables (see, for example, column 15, lines 34-38). In one example, the calls to the procedures are not omitted from the consolidated loop code segment (FIG. 12) because the calls do result in changes to the program variables (FIG. 11). Johnson discloses another example in which the function *f* of argument *x* is invariant and omitted from the consolidated loop code segment (column 8, lines 51-64).

Mahadevan also discloses the steps:

(4) determining whether the consolidated code segment should be executed in preference to the non-optimized loop code segments (see, for example, column 10, lines 46-52, which shows determining the most favorable optimization); and

(5) if said determination is favorable, including the consolidated code segment in optimized code for a program written in the high level programming language (see, for example, column 10, lines 53-61, which shows optimizing the code using the most favorable optimization).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Mahadevan with the features taught by Hirai and Johnson, wherein the non-optimized loop code segment includes a call to a procedure that invokes the procedure responsive to arguments of the call, the arguments including ones of the

program variables, and wherein the consolidated code includes certain code of the non-optimized loop code segment but omits the call to the procedure if certain program variables are not changed. The combination would have been obvious because one of ordinary skill in the art would have been motivated to omit invariant code from the loop code segment to improve execution speed (see, for example, Johnson, column 8, lines 65-67), even when the loop code segment includes calls to procedures (see, for example, Hirai, column 6, lines 6-13), so that the method of Mahadevan could save additional CPU cycles.

With respect to claim 2 (currently amended), Mahadevan discloses a method of optimizing the compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs (see, for example, the abstract), the method comprising the steps:

(1) providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct is executed a loop repetition number of times *n* (see, for example, column 6, lines 22-27, which shows providing loop code for a loop written in a high-level language, and column 7, lines 30-35, which shows that the loop is iteratively executed some number of times, i.e. *n* times).

Mahadevan does not expressly disclose the limitation wherein the non-optimized loop code segment includes a call to a procedure.

However, Hirai discloses an analogous method of optimizing a loop code segment (see, for example, the abstract), and discloses a non-optimized loop code segment that includes a call

to a procedure (see, for example, loop block 1105 in FIG. 11, which includes calls 1108 and 1109 to procedures).

Mahadevan also discloses the steps:

(2) providing a non-optimized pre-loop code segment corresponding to programming instructions preceding the loop construct, and a non-optimized post-loop code segment corresponding to instructions succeeding the loop construct (see, for example, FIG. 3, which shows instructions preceding and succeeding the loop to be provided as pre-loop and post-loop code, respectively);

(3) providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for causing execution of the loop construct the loop repetition number of times *n* (see, for example, column 7, lines 35-39, which shows providing a variable that has values satisfying initial and terminating conditions for executing the loop *n* times).

Although Hirai discloses that the procedure includes ones of the program variables (see, for example, procedure 1118 in FIG. 11, which shows that the procedure includes ones of the program variables 1102 used in the loop block 1105), Mahadevan and Hirai do not expressly disclose the limitation wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables.

However, Johnson discloses an analogous method of optimizing a loop code segment (see, for example, column 7, lines 20-27), and suggests that a call to a procedure, such as in Hirai, could invoke the procedure with arguments that include ones of the program variables used in the loop code segment (see, for example, column 7, lines 28-40, which shows an

argument *x* to some function *f*, where *x* is a variable that has values satisfying initial and terminating conditions for the loop).

Mahadevan also discloses the steps:

(4) revising the non-optimized pre-loop, loop and post-loop code segments to include the execution conditions (see, for example, FIG. 3, which shows including the execution conditions with the pre-loop, loop and post-loop code); and

(5) optimizing the non-optimized pre-loop, loop and post-loop code segments for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n* (see, for example, column 10, line 53 to column 11, line 8, which shows optimizing the loop to provide code corresponding to the trip count, i.e. the number of repetitions *n*, and optimizing the placement of compensation code as pre-loop or post-loop code; also see, for example, FIGS. 5 and 6, which show optimized pre-loop, loop and post-loop code).

Although Mahadevan discloses that branches are omitted from the consolidated loop code segment to save CPU cycles and reduce branch mispredictions (see, for example, column 6, lines 28-32), Mahadevan does not expressly disclose the limitation wherein the consolidated code includes certain code of the non-optimized loop code segment and omits certain other code of the non-optimized loop code segment and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables.

However, Hirai further discloses that the consolidated loop code segment includes certain code of the non-optimized loop code segment and omits other invariant code (see, for example,

FIG. 7 and column 23, lines 61-67), which is omitted if the call invoking the procedure results in no change in the program variables (see, for example, column 15, lines 34-38). In one example, the calls to the procedures are not omitted from the consolidated loop code segment (FIG. 12) because the calls do result in changes to the program variables (FIG. 11). Johnson discloses another example in which the function *f* of argument *x* is invariant and omitted from the consolidated loop code segment (column 8, lines 51-64).

Mahadevan also discloses the steps:

(6) determining whether the consolidated code segment should be executed in preference to the non-optimized code segments (see, for example, column 10, lines 46-52, which shows determining the most favorable optimization); and

(7) if said determination is favorable, including the consolidated code segment in optimized code for a program written in the high level programming language (see, for example, column 10, lines 53-61, which shows optimizing the code using the most favorable optimization).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the method of Mahadevan with the features taught by Hirai and Johnson, wherein the non-optimized loop code segment includes a call to a procedure that invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables, and wherein the consolidated code includes certain code of the non-optimized loop code segment but omits the call to the procedure if certain program variables are not changed. The combination would have been obvious because one of ordinary skill in the art would have been motivated to omit invariant code from the loop code segment to improve

execution speed (see, for example, Johnson, column 8, lines 65-67), even when the loop code segment includes calls to procedures (see, for example, Hirai, column 6, lines 6-13), so that the method of Mahadevan could save additional CPU cycles.

With respect to claims 3 and 7 (previously presented), Mahadevan also discloses the limitation wherein said determination involves a cost-benefit analysis to determine whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment (see, for example, column 10, lines 46-52, which shows determining the most favorable optimization while taking into account the cost of not using the most favorable optimization, and column 10, lines 7-45, which shows thresholds used in the determination).

With respect to claims 4 and 8 (original), Mahadevan also discloses the limitation wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions (see, for example, column 9, lines 46-50, which shows that the inclusion of the optimized loop in the code is conditional, e.g. on the occurrence of the execution conditions).

With respect to claims 5 and 9 (original), Mahadevan also discloses the limitation wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops (see, for example, column 2, lines 22-24, which shows that the loops may include for loops, while loops and do loops, i.e. repeat loops).

With respect to claim 6 (original), Mahadevan also discloses the limitation wherein said steps (1) to (5) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1 (see, for example, column 7, lines 30-35, which shows that the operations are repeated for each iteration of the loop, and column 6, lines 22-27, which shows processing all the loops).

With respect to claim 10 (original), Mahadevan also discloses the limitation wherein said steps (1) to (7) are repeated a predetermined number of times *k*, for values of the loop repetition number *n* from 0 to *k*-1 (see, for example, column 7, lines 30-35, which shows that the operations are repeated for each iteration of the loop, and column 6, lines 22-27, which shows processing all the loops).

With respect to claim 11 (currently amended), Mahadevan discloses a compiler for optimizing the compiled code generated from high level computer programming languages, wherein the compiled code includes loop constructs, the compiler being embodied on a computer-readable medium (see, for example, column 11, lines 9-22, which shows a compiler for optimizing code with loops generated from a high-level language, and FIG. 1, which shows an associated computer-readable medium). The additional features and limitations of this claim are analogous to the limitations recited in claim 1 (see the rejection of claim 1 above).

With respect to claim 12 (currently amended), Mahadevan discloses a compiler for optimizing the compiled code generated from high level computer programming languages wherein the compiled code includes loop constructs, the compiler being embodied on a computer-readable medium (see, for example, column 11, lines 9-22, which shows a compiler

for optimizing code with loops generated from a high-level language, and FIG. 1, which shows an associated computer-readable medium). The additional features and limitations of this claim are analogous to the limitations recited in claim 2 (see the rejection of claim 2 above).

With respect to claims 13 and 17 (previously presented), the limitations of these claims are analogous to the limitations recited in claims 3 and 7 (see the rejections of claims 3 and 7 above).

With respect to claims 14 and 18 (original), the limitations of these claims are analogous to the limitations recited in claims 4 and 8 (see the rejections of claims 4 and 8 above).

With respect to claims 15 and 19 (original), the limitations of these claims are analogous to the limitations recited in claims 5 and 9 (see the rejections of claims 5 and 9 above).

With respect to claim 16 (original), the limitations of this claim are analogous to the limitations recited in claim 6 (see the rejection of claim 6 above).

With respect to claim 20 (original), the limitations of this claim are analogous to the limitations recited in claim 10 (see the rejection of claim 10 above).

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Yigdall whose telephone number is (571) 272-3707. The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

MY

Michael J. Yigdall
Examiner
Art Unit 2192

mjy

TUAN DAM
SUPERVISORY PATENT EXAMINER